

PATENT ABSTRACTS OF JAPAN 4

(11)Publication number : 07-200315

(43)Date of publication of application : 04.08.1995

(51)Int.Cl.

G06F 9/46

(21)Application number : 06-012101

(71)Applicant : FUJI FACOM CORP

(22)Date of filing : 07.01.1994

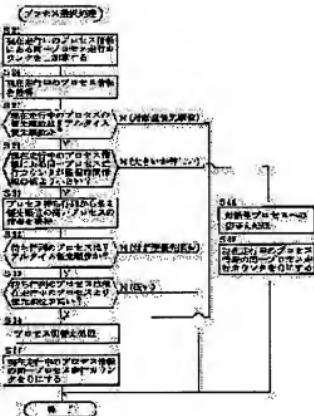
(72)Inventor : YAMAGUCHI HIROSHI

(54) SCHEDULING DEVICE FOR PROCESS IN COMPUTER SYSTEM

(57)Abstract:

PURPOSE: To attain proper scheduling by preventing a CPU from being continuously occupied by real time process.

CONSTITUTION: A same process running counter is incremented by 1 (S21), information relating to a process running at present is acquired from process information running at present (S26) to discriminate whether or not the process running at present has priority in real time. In the case of the priority of real time (S27Y), a count of a same process running counter in the process information running at present is compared with monitor time information and when the count of the same process running counter is larger than or equal to the monitor time information value (S28N), an interactive process changeover module is executed (S44), and the same process running counter in the process information running at present is set to zero and the program is terminated (S47).



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平7-200315

(43)公開日 平成7年(1995)8月4日

(51)Int.Cl.
G 0 6 F 9/46識別記号 庁内整理番号
3 4 0 B 7629-5B

F I

技術表示箇所

(21)出願番号

特願平6-12101

(22)出願日

平成6年(1994)1月7日

(71)出願人 000237156

富士ファコム制御株式会社
東京都日野市富士町1番地(72)発明者 山口 弘
東京都日野市富士町1番地 富士ファコム
システム株式会社内

(74)代理人 弁理士 森田 越一

(54)【発明の名称】 計算機システムにおけるプロセスのスケジューリング装置

(57)【要約】

【目的】 リアルタイムプロセスによりCPUが連続して占有されることを防ぎスケジューリングを適正にする。

【構成】 同一プロセス走行カウンタに1を加算してから(S21)、現在走行中のプロセスに関する情報を現在走行中プロセス情報から獲得し(S26)、現在走行中のプロセスがリアルタイムの優先順位か否かを判定する。リアルタイムの優先順位である場合は(S27Y)、さらに、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、同一プロセス走行カウンタの値が監視時間情報の値よりも大きいか等しい場合は(S28N)、対話型プロセス切替えモジュールを実行し(S44)、現在走行中プロセス情報の同一プロセス走行カウンタを0にして終了する(S47)。



【特許請求の範囲】

【請求項1】 プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型プロセスの中の優先順位が最高の対話型プロセスにCPU占有を割り当てる手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項2】 プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、

同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項3】 プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスを強制終了するとともにプロセス待ち行列のリンクから外す手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項4】 プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間

で互いの優先順位を入れ換える手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項5】 プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

リアルタイムクラスのプロセスが連続してCPUを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型クラスの中の優先順位が最高の対話型プロセスにCPU占有を割り当てる手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項6】 プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、

リアルタイムクラスのプロセスが連続してCPUを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、1つのオペレーティングシステムで、対話型プロセスとリアルタイムプロセスに対し、定期周期タイム割込みによりCPUを割り当てる際のスケジューリング装置に関する。

【0002】

【従来の技術】 従来、計算機システムにおいて、1つのオペレーティングシステム上に、リアルタイムプロセスと対話型プロセスとが混在することがある。ここでいうリアルタイムプロセスは、応答時間が厳しく制限されたプロセスであり、所定の時間内に処理が完了するため処理時間を予測することが可能である。また、対話型プロセスは、応答時間の制限は厳しくないが、どのプロセスにも均等にCPUが割り当てられる。

【0003】 さらに詳しく述べると、リアルタイムプロセスが複数のプロセスにより構成される場合、それぞれのプロセスはCPUを割り当てるための優先順位が定められており、優先順位が高いプロセスから順にCPUの占有権を獲得する。また、CPUを占有したプロセスは

処理が完了するまで占有権を放棄しない。すなわち、あるプロセスがCPUを占有している間は、このプロセスより優先順位が下位のプロセスはCPUの割当を待つ状態を続ける。しかし、優先順位が高いプロセスに処理要求があると、そのプロセスはCPUを占有しているプロセスからCPUの占有権を奪い取り、自身がCPUを占有する。このようにして、優先順位が高いプロセスから順に処理される。なお、この優先順位は、ユーザにより指定されるものである。

【0004】一方、対話型プロセスは、オペレーティングシステムによって各プロセスの優先順位が決められる。すなわち、計算機システムの起動時は、各プロセスの優先順位はどれも同じであるが、各プロセスがCPUを占有することに、そのプロセスの優先順位が低位に変更される。それにより、CPUの占有回数の少ないプロセスの優先順位が相対的に高くなる、順に最高となったプロセスからCPUを占有することになる。その結果、平均すると、各プロセスに対してCPUの割当時間が均等になる。

【0005】なお、このような対話型プロセスを有する計算機システムでは、一定間隔ごとの定期窓タイム割込みがオペレーティングシステムとしておこなわれ、この割込み間隔を1単位時間とすると、対話型プロセスは1単位時間でCPUの割当で切り替わる。すなわち、ある対話型プロセスが1単位時間CPUを占有したら、そのプロセスは優先順位を低位に変更したのちCPUの占有権を放棄する。次いで、その次の優先順位の対話型プロセスがCPUの占有権が引き渡される。これらはリアルタイムプロセスと対話型プロセスとが1つのオペレーティングシステム上に混在する計算機システムにおけるスケジューリング方法は、次のように行われている。

【0006】リアルタイムプロセスと対話型プロセスには、両プロセスごとに異なる優先順位を持たせる。例えば、システムに128の優先順位がある場合に、優先順位0~63をリアルタイムプロセスとし、残りの優先順位64~127を対話型プロセスとする。この優先順位は数字が小さい程、優先順位が高くなる。また、実行可能であってCPUが割当られてないプロセスをリンクしておくために、プロセスの優先順位ごとに待ち行列を設ける。それにより、CPUを占有をしていたプロセスが占有権を放棄した場合、この待ち行列のなかの優先順位が最高位のプロセスがCPUの占有権を獲得する。なお、優先順位の同じプロセスが複数個あった場合は、到着順(FCFS: First Come First Service)によりCPUの占有権を得る。

【0007】リアルタイムプロセスと対話型プロセスとの間でのCPUの占有については、リアルタイムプロセスを優先し、全てのリアルタイムプロセスの処理が完了してから、対話型プロセスにCPUの占有権を与える。また、対話型プロセスがCPUの占有権を得ている間で

あっても、新たにリアルタイムプロセスの処理要求が発生すると、対話型プロセスはCPU占有権は、リアルタイムプロセスに奪い取られる。これらの処理を行う従来のオペレーティングシステムの構成を示したのが図14の概念図であり、スケジューリング処理を系統的に示したのが図11、図12、図13のフローチャートである。

【0008】図14に示すように、オペレーティングシステムはタイマ割込みが処理モジュール、プロセス選択処理モジュール、プロセス切替え処理モジュール、プロセス待ち行列および現在走行中のプロセス情報から構成されており、定期窓タイム割込みが発生すると、タイマ割込み処理が呼び出される。図11は、タイマ割込み処理モジュールを示す。このモジュールは、先ず、所定のタイマ処理を行い、次にプロセス選択処理モジュールへ進む。

【0009】図12は、プロセス選択処理モジュールの内容を示す。このモジュールは、先ず、現在走行中のプロセスに関する情報を獲得し(S26)、次いで、現在走行中のプロセスがリアルタイムの優先順位か否かを判別する(S27)。リアルタイムの優先順位である場合は(S27Y)、プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する(S31)。次に、待ち行列のプロセスはリアルタイム優先順位か否かを判別し(S32)、リアルタイム優先順位である場合(S32Y)は、さらに、待ち行列のプロセスは現在走行中のプロセスより優先順位が高いか否かを判別する(S33)。

【0010】優先順位が高い場合(S33Y)、すなわち図14のプロセス行列待ちに、現在走行中のプロセスより優先順位の高いリアルタイムプロセスがリンクされていたらプロセス切替え処理モジュール(S34)へ進む。待ち行列のプロセスがリアルタイム優先順位でなく対話型優先順位である場合(S32N)および優先順位が高くない低い場合(S33N)は、現在走行中のプロセスがそのままCPUの占有を続ける。また、S27でリアルタイム優先順位でない場合も、プロセス切替え処理モジュール(S34)へ進む。

【0011】図13は、プロセス切替え処理モジュールの内容を示す。このモジュールは、先ず、プロセス待ち行列に現在走行中のプロセスをリンクする(S11)。なお、ここで現在走行中のプロセスが対話型プロセスだったら、優先順位を再計算してからリンクする。これは、対話型プロセスはCPUを占有するごとに優先順位を下げることで、CPUの割当の少ないプロセスの優先順位を相対的に高めるためである。次に、プロセス待ち行列から最も優先順位が高いプロセスを選びだしプロセス待ち行列のリンクから外し(S12)、CPUを割り当てる(S13)。最後に、現在走行中のプロセス情報(図14)にCPUを新たに割当たプロセスの識別

子と優先順位を設定する(S14)。

【0012】

【発明が解決しようとする課題】しかしながら、上述したオペレーティングシステムのスケジューリング方法では、例えば、リアルタイムプロセスが何らかの原因でCPUの占有権を放棄しなかったり、処理が所定時間内に完了しなかったりした場合に、計算機システムにデッドロックが発生したり、対話型プロセスが全く動作しなくなったり、応答時間が異常に遅くなる等の問題が生じた。本発明は上記問題点を解決するためになされたもので、その目的とするところは、リアルタイムプロセスが連続してCPUを占有することを防ぐことにより動作を安定させることができると計算機システムにおけるプロセスのスケジューリング装置を提供することにある。

【0013】

【課題を解決するための手段】上記目的を達成するため、第1の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型プロセスの中の優先順位が最高の対話型プロセスにCPU占有を割り当てる手段とを備えたことを特徴とする。

【0014】第2の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段とを備えたことを特徴とする。

【0015】第3の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスを強制終了するとともにプロセス待ち行列のリンクから外す手段とを備えたことを特徴とする。

【0016】第4の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによ

りCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、同一のリアルタイムプロセスが連続してCPUを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間で互いの優先順位を入れ換える手段とを備えたことを特徴とする。

【0017】第5の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、リアルタイムクラスのプロセスが連続してCPUを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型クラスの中の優先順位が最高の対話型プロセスにCPU占有を割り当てる手段とを備えたことを特徴とする。

【0018】第6の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定期周期タイム割込みによりCPU占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、リアルタイムクラスのプロセスが連続してCPUを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段とを備えたことを特徴とする。

30 【0019】

【作用】第1の発明においては、同一のリアルタイムプロセスが連続してCPUを占有した回数がカウントされそのカウント値が所定値に達すると、プロセス待ち行列にリンクされている対話型プロセスの中の優先順位が最高の対話型プロセスにCPU占有が割り当られる。それにより、同一のリアルタイムプロセスが無制限に連続してCPUを占有することがなくなる。

【0020】第2の発明においては、同一のリアルタイムプロセスが連続してCPUを占有した回数がカウントされそのカウント値が所定値に達すると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングが実行される。それにより、同一のリアルタイムプロセスが連続してCPUを占有した場合にユーザが任意の対処を行なうことが可能になる。

【0021】第3の発明においては、同一のリアルタイムプロセスが連続してCPUを占有した回数がカウントされそのカウント値が所定値に達すると、現在走行中のリアルタイムプロセスが強制終了されるとともにそのリアルタイムプロセスがプロセス待ち行列のリンクから外される。それにより、異常原因等により連続してCPU

50

を占有したリアルタイムプロセスが除去され、以後正常な動作に復帰する。

【0022】第4の発明においては、同一のリアルタイムプロセスが連続してCPUを占有した回数がカウントされそのカウント値が所定値に達すると、現在走行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間で互いの優先順位が入れ替わる。それにより、同一のリアルタイムプロセスが無制限に連続してCPUを占有することができなくなる。

【0023】第5の発明においては、リアルタイムクラスのプロセスが連続してCPUを占有した回数がカウントされそのカウント値が所定値に達すると、プロセス待ち行列にリンクされている対話型クラスの上の優先順位が最高の対話型プロセスにCPU占有が割り当てられる。それにより、リアルタイムプロセスが無制限に連続してCPUを占有することができなくなる。

【0024】第6の発明においては、リアルタイムクラスのプロセスが連続してCPUを占有した回数がカウントされそのカウント値が所定値に達すると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングが実行される。それにより、リアルタイムプロセスが連続してCPUを占有した場合にユーザが任意の対応を行うことが可能になる。

【0025】

【実施例】以下、図に沿って本発明の実施例を説明する。この実施例で対象とする計算機システムのオペレーティングシステムは、プロセスの優先順位に基づき、対話型プロセスとリアルタイムプロセスのCPUへの割り当てを管理するものであり、以下の条件で作動する。

(1) 定周期タイマが一定間隔(例えは、10ms)でオペレーティングシステムに割込みを発生する。

(2) 対話型プロセスとリアルタイムプロセスは互いに他者に対し、異なる優先順位を持つ。つまり、同種のプロセス間では同じ優先順位を持つことがあっても、対話型プロセスとリアルタイムプロセスと同じ優先順位を持つことはない。

(3) CPUの割当てを待つためのプロセスの待ち行列を、優先順位ごとに持つ。

(4) 各プロセスは、システムで唯一の識別子を持つ。

【0026】図1は本発明が適用されるオペレーティングシステムの構成を示す概念図である。図示されるように、オペレーティングシステムはタイマ割込み処理モジュール、プロセス選択処理モジュール、プロセス切替え処理モジュール、対話型プロセス切替え処理モジュール、プロセス待ち行列、現在走行中プロセス情報、監視時間情報、ハンドラプログラム登録テーブルはリアルタイムプロセスが所定時間以上CPUを占有したときに実行されるハンドラプログラムのエンタリードレスを格納している。このハンドラプログラムはユーザによって予め登録される。リアルタイムプロセス抑止フラグはリアルタイムクラスのプロセスのCPUへの割当てを抑止する。

從来技術の項で示した図14の動作と同じであるのでその詳細な説明は省略する。また、プロセス選択処理モジュールおよび対話型プロセス切替え処理モジュールが本発明の主なる特徴であり、詳細な動作は後述する。

【0027】プロセス待ち行列は、プロセスごとの優先順位に対応したリンクターミナルを備え、CPUの割当てを待つリアルタイムプロセスおよび対話型プロセスをそれぞれ対応する優先順位のリンクターミナルにリンクする。図示例では、優先順位が0～127の128あり、優先順位は数値が小さいほど高くなる。このうち上位の0～63の優先順位をリアルタイムクラスとしてリアルタイムプロセスの優先順位とする。また残り下位の64～127の優先順位を対話型クラスとして対話型プロセスの優先順位とする。

【0028】現在走行中プロセス情報は、プロセスの識別子、プロセスの優先順位、同一プロセス走行カウンタおよび同一クラスプロセス走行カウンタからなる。このプロセス識別子には、現在CPUが割当でられているプロセスの識別子が格納される。プロセスの優先順位には、現在CPUが割当でられているプロセスの優先順位が格納される。同一プロセス走行カウンタは、同一のプロセスに対して連続してCPUが割り当てられた時間が単位時間(定期周期タイマ割込み1回あたりのCPU占有時間)の個数として格納され、CPUが異なるプロセスへ割り当てられたとき、このカウンタはゼロクリアされる。

【0029】同一クラスプロセス走行カウンタは、同一のクラスのプロセスに対して連続してCPUが割り当てられた時間が単位時間の個数として格納され、CPUが異なるクラスのプロセスへ割り当てられときとこのカウンタはゼロクリアされる。

【0030】同一クラスプロセス走行カウンタは、リアルタイムプロセスが所定時間以上CPUを占有していないかどうかをチェックする際の基準値として用いられる。つまり、この監視時間情報に予めシステムに応じた値として上述した単位時間の個数nをセットしておき、同一プロセス走行カウンタまたは同一クラスプロセス走行カウンタのカウント値と監視時間情報の値を比較し、カウント値が監視時間情報の値に達したら異常事態の発生と見なす。

【0030】ハンドラプログラム登録テーブルはリアルタイムプロセスが所定時間以上CPUを占有したときに実行されるハンドラプログラムのエンタリードレスを格納している。このハンドラプログラムはユーザによって予め登録される。リアルタイムプロセス抑止フラグはリアルタイムクラスのプロセスのCPUへの割当てを抑止する。

【0031】次に、各発明をフローチャートに基づき説明する。図2は、第1の発明に係るプロセス選択処理モジュールの第1の実施例を示すフローチャートである。このモジュールは、まず、現在走行中のリアルタイムプロセスが連続してCPUを占有している時間を知るた

め、現在走行中のプロセス情報にある同一プロセス走行カウンタに1を加算する(S21)。次に、現在走行中のプロセスに関する情報を現在走行中プロセス情報から獲得し(S26)、次いで、現在走行中のプロセスがリアルタイムの優先順位か否かを判別する(S27)。リアルタイムの優先順位である場合は(S27Y)、さらに、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さい場合は(S28Y)、プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する(S31)。

【0032】次に、待ち行列のプロセスがリアルタイム優先順位か否かを判別し(S32)、リアルタイム優先順位である場合は(S32Y)、さらに、待ち行列のプロセスは現在走行中のプロセスより優先順位が高いか否かを判別する(S33)。優先順位が高い場合(S33Y)、すなわち図1のプロセス行列待ちに、現在走行中のプロセスより優先順位の高いリアルタイムプロセスがリンクされいたら図1のプロセス切換え処理モジュールを実行し(S34)、次いで、現在走行中プロセス情報の同一プロセス走行カウンタを0にして終了する(S37)。

【0033】また、S27でリアルタイム優先順位でない場合は、S34のプロセス切換え処理モジュールへ進む。さらに、S28で同一プロセス走行カウンタの値が監視時間情報の値よりも大きい場合は、図3の対話型プロセス切替えモジュールを実行し(S44)、次いで、現在走行中プロセス情報の同一プロセス走行カウンタを0にして終了する(S47)。またさらに、S32で待ち行列のプロセスがリアルタイム優先順位でなく対話型優先順位である場合、およびS33で優先順位が高くない、つまり低い場合は処理を終了することにより、現在走行中のプロセスにそのままCPUの占有を続けさせる。

【0034】図3は、対話型プロセス切替えモジュールを示すフローチャートである。先ず、対話型クラスのプロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する(S61)。次に、得られたプロセスは対話型プロセスであるか否かを判別し、対話型プロセスでない場合は(S62N)、処理を終了する。対話型プロセスである場合は(S62Y)、現在走行中のプロセスをプロセス待ち行列へリンクする(S63)。なお、ここでリンクするプロセスが対話型プロセスである場合は、優先順位を再計算してからリンクする。

【0035】次に、対話型クラスのプロセス待ち行列から最も優先順位の高いプロセスのリンクを外し(S64)、次いで、待ち行列からリンクが外されたプロセスにCPUを割り当てる(S65)、同時に現在走行中プロセス情報に、CPUを割り当てるプロセスの識別子と優先順位を設定して終了する(S66)。これらの第1の実施例にかかる処理では、S28、S44の処理を実行

することで、同一のリアルタイムプロセスによりCPUが連続して占有される場合でも、同一プロセス走行カウンタにセットされたn回のCPU占有がなされた後に、対話型プロセスによるCPUの占有が1回行われる。すなわち、n+1回の定期割込みの間に少なくとも1回は対話型プロセスが実行されることになる。

【0036】その結果、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくとも、計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなることがなくなる。また、対話型プロセスから暴走等した同一のリアルタイムプロセスを強制終了させることもできる。さらに、同一のリアルタイムプロセスが暴走等しても、それを異常とみなさずそのままリアルタイム処理を継続させたいというようなシステムにこの実施例は有効である。

【0037】図4は、第1の発明に係るプロセス選択処理モジュールの第2の実施例を示すフローチャートである。このフローチャートは、図2のフローチャートにS22、S41、S42を追加したものであり、他は図2と共に並んであるので共通部分の説明を省略して異なる部分についてのみ説明する。すなわち、S21で、現在走行中のプロセス情報にある同一プロセス走行カウンタに1を加算した後、リアルタイムプロセス抑止フラグをチャックし、1がセットされていれば(S22Y)、そのままS26へ進み、1がセットされていなければ(S22N)、S41へ進む。

【0038】また、S28で現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合(S28N)、S41へ進む。S41では、リアルタイムプロセス抑止フラグをチャックし、1がセットされていれば(S41N)、そのままS44へ進み、1がセットされていなければ(S41Y)、S42へ進み、リアルタイムプロセス抑止フラグに1をセットして、S44へ進む。これら第2の実施例にかかる処理では、同一のリアルタイムプロセスによりCPUが連続して占有されると、リアルタイムプロセス抑止フラグに1がセットされて対話型プロセスにCPUの制御権が渡され、以後毎回とも図3の対話型プロセス切替えモジュールが実行されることになる。

【0039】その結果、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくとも、計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなることがなくなる。また、対話型プロセスから暴走等した同一のリアルタイムプロセスを強制終了させることもできる。さらに、同一のリアルタイムプロセスが暴走等したら、システムに異常が発生したものとみなし、リアルタイム処理をそのまま続行せずに何らかの原因調査を行いたいというようなシステムにこの実施例は有効である。

【0040】図5は、第2の発明に係るプロセス選択処理モジュールの第3の実施例を示すフローチャートである。このフローチャートは、図2のS44をS45、S46にかえたものであり、他は図2と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、S28で、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合(S28N)、S45へ進む。

【0041】S45では、ハンドラプログラム登録テーブルにハンドラプログラムのエンティアドレスが登録されているか否かを判別し、登録されていなければ(S45N)、そのままS47へ進む。登録されていれば(S45Y)、そのエンティアドレスからハンドラプログラムを呼び出して実行し(S46)、S47へ進む。これら第3の実施例にかかる処理では、同一のリアルタイムプロセスによりCPUがn回連続して占有されると、ハンドラプログラムが呼びだされて実行される。このハンドラプログラムは予めユーザにより登録されているため、CPUの割当時のスケジューリングをユーザが自由に設定してユーザ固有の最適なシステムを構築することができる。

【0042】図6は、第3の発明に係るプロセス選択処理モジュールの第4の実施例を示すフローチャートである。このフローチャートは、図2のS44、S47をS48にかえたものであり、他は図2と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、S28で、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合(S28N)、S48へ進む。S48では、現在走行中のリアルタイムプロセスのCPU占有を強制終了する。このとき、終了したリアルタイムプロセスを、待ち行列のリンクタームナルへ接続させない。次に、S34のプロセス切換え処理モジュールを実行することにより、次の優先順位のプロセスにCPUが割り当たられる。

【0043】これら第4の実施例にかかる処理では、同一のリアルタイムプロセスによりCPUがn回連続して占有されると、それまで走行中のリアルタイムプロセスが強制的に終了され、次に優先順位の高いプロセスにCPUが割当されることで同一リアルタイムプロセスによるCPUの独占が回避される。その結果、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくとも、計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなったりすることがなくなる。

【0044】もっとも、対話型プロセスが動作するは、リアルタイムプロセスが終了してからである。また、対話型プロセスから暴走等した同一のリアルタイムプロセスを強制終了させることもできる。さらに、同一のリアルタイムプロセスが暴走等したら、そのプロセス

だけに異常が発生したとみなしてリアルタイム処理を維持させたいというようなシステムにこの実施例は有効である。

【0045】図7は、第4の発明に係るプロセス選択処理モジュールの第5の実施例を示すフローチャートである。このフローチャートは、図2のS44、S47をS49へS52にかえたものであり、他は図2と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、S28で、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合(S28N)、S49へ進む。S49では、リアルタイムプロセスの待ち行列から最も優先順位の高いプロセスの情報を獲得する。

【0046】次いで、獲得した情報はリアルタイムプロセスか否かを判別し、リアルタイムプロセスでない場合は(S50N)、処理を終了する。リアルタイムプロセスである場合は(S50Y)、現在走行中のリアルタイムプロセスとリアルタイムクラスのプロセス待ち行列の中で最も優先順位の高いプロセスとの互いの優先順位を交換する(S51)。次に、現在走行中のリアルタイムプロセスをリアルタイムクラスのプロセス待ち行列にリンクし(S52)、次いで、S34のプロセス切換え処理モジュールを実行することにより、次の優先順位のリアルタイムプロセスにCPUが割り当たる。

【0047】これら第5の実施例にかかる処理では、同一のリアルタイムプロセスによりCPUがn回連続して占有されると、それまで走行中のリアルタイムプロセスと次に優先順位の高いリアルタイムプロセスとの優先順位を交換し、それまでの優先順位であったリアルタイムプロセスにCPUが割り当たることで、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくとも計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなったりすることがなくなる。

【0048】もっとも、対話型プロセスが動作するは、リアルタイムプロセスが終了してからである。また、同一のリアルタイムプロセスが暴走等しても、それを異常とみなさず、しかも対話型プロセスにCPUの占有権を渡さずに、リアルタイム処理は継続したいというようなシステムにこの実施例は有効である。また、この実施例の場合、暴走等した同一のリアルタイムプロセスを強制終了させようとすると、次に優先順位の高いプロセスの処理が重要となる。

【0049】図8は、第5の発明に係るプロセス選択処理モジュールの第6の実施例を示すフローチャートである。このモジュールは、先ず、現在走行中のリアルタイムクラスのプロセスが連続してCPUを占有している時間を知るため、現在走行中のプロセス情報にある同一クラスプロセス走行カウンタに1を加算する(S23)。次に、現在走行中のプロセスに関する情報を現在走行中

プロセス情報から獲得し(S26)、次いで、現在走行中のプロセスがリアルタイムの優先順位か否かを判別する(S27)。

【0050】リアルタイムの優先順位である場合は(S27Y)、さらに、現在走行中プロセス情報中の同一クラスプロセス走行カウンタの値を監視時間情報の値と比較し、小さい場合は(S29Y)、プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する(S31)。次に、待ち行列のプロセスがリアルタイム優先順位か否かを判別し(S32)、リアルタイム優先順位である場合は(S32Y)、さらに、待ち行列のプロセスは現在走行中のプロセスより優先順位が高いか否かを判別する(S33)。

【0051】優先順位が高い場合(S33Y)、すなわち図1のプロセス行列待ちに、現在走行中のプロセスより優先順位の高いリアルタイムプロセスがリンクされていたら図1のプロセス切換え処理モジュールを実行する(S34)。ここでさらに、以前走行していたプロセスと現在走行中のプロセスのクラスが違うか否かを判別し、違う場合は(S35Y)、現在走行中プロセス情報の同一クラスプロセス走行カウンタを0にして終了する(S37)。また、S27でリアルタイム優先順位でない場合は、プロセス切換え処理モジュール(S34)へ進む。

【0052】さらに、S29で同一クラスプロセス走行カウンタの値が監視時間情報の値よりも大きいか等しい場合は、図3の対話型プロセス切替えモジュールを実行し(S44)、以前走行していたプロセスと現在走行中のプロセスのクラスが違うか否かを判別し、違う場合は(S53Y)、現在走行中プロセス情報の同一クラスプロセス走行カウンタを0にして終了する(S54)。また、プロセスのクラスが同じ場合は(S53N)、そのまま終了する。またさらに、S32で待ち行列のプロセスがリアルタイム優先順位でなく対話型優先順位である場合、S33で優先順位が低い場合およびS35で以前走行していたプロセスと現在走行中のプロセスのクラスが同じ場合は処理を終了することにより、現在走行中のプロセスがそのままCPUの占有を続ける。

【0053】これらの第6の実施例にかかる処理では、S29、S44の処理を実行することで、リアルタイムクラスのプロセスによりCPUが連続して占有される場合でも、同一クラスプロセス走行カウンタにセットされたn回のCPU占有がなされた後に、対話型プロセスによるCPUの占有が1回行われる。すなわち、n+1回の定期割込みの間に少なくとも1回は対話型プロセスが実行されることになる。また、第6の実施例は、第1の実施例とほぼ同じ効果が得られるとともに、対話型プロセスの応答時間が異常に遅くなることも解消される。

【0054】図9は、第5の発明に係るプロセス選択処理モジュールの第7の実施例を示すフローチャートであ

る。このフローチャートは、図8のフローチャートにS24、S41、S42を追加したものであり、他は図8と共通であるので共通部分の説明を省略して異なる部分についてのみ説明する。すなわち、S23で、同一クラスプロセス走行カウンタに1を加算した後、リアルタイムプロセス抑止フラグをチェックし、1がセットされていれば(S24Y)、そのままS26へ進み、1がセットされていなければ(S24N)、S41へ進む。また、S29で現在走行中プロセス情報中の同一クラスプロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合(S29N)、S41へ進む。

【0055】S41では、リアルタイムプロセス抑止フラグをチェックし、1がセットされていなければ(S41N)、そのままS44へ進む。1がセットされていなければ(S41Y)、S42へ進み、リアルタイムプロセス抑止フラグに1をセットして、S44へ進む。これら第7の実施例にかかる処理では、リアルタイムクラスのプロセスによりCPUがn回連続して占有されると、リアルタイムプロセス抑止フラグに1がセットされて、以後は毎回も図3の対話型プロセス切替えモジュールが実行されることになる。また、第7の実施例は、第2の実施例とはほぼ同じ効果が得られるとともに、対話型プロセスの応答時間が異常に遅くなることも解消される。

【0056】図10は、第6の発明に係るプロセス選択処理モジュールの第8の実施例を示すフローチャートである。このフローチャートは、図8のS44をS45、S46にかえたものであり、他は図8と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、S29で、現在走行中プロセス情報中の同一クラスプロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合(S29N)、S45へ進む。S45では、ハンドラプログラム登録テーブルにハンドラプログラムのエンティアドレスが登録されているか否かを判別し、登録されていなければ(S45N)、そのままS53へ進む。登録されていなければ(S45Y)、そのエンティアドレスからハンドラプログラムを呼び出して実行して(S46)、S53へ進む。

【0057】これら第8の実施例にかかる処理では、リアルタイムクラスのプロセスによりCPUがn回連続して占有されると、ハンドラプログラムを呼びだして実行される。このハンドラプログラムは予めユーザにより登録されているため、CPUの割当でのスケジューリングをユーザが自由に設定してユーザ固有の最適なシステムを構築することができる。また、第8の実施例は、第3の実施例とほぼ同じ効果が得られるとともに、対話型プロセスの応答時間が異常に遅くなることも解消される。

【0058】

【発明の効果】以上述べたように第1の発明によれば、同一のリアルタイムプロセスが連続してCPUを占有した回数が所定回数になると、プロセス待ち行列にリンク

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

されている対話型プロセスの中の優先順位が最高の対話型プロセスにCPUの占有を割り当てることにより、同一のリアルタイムプロセスが無制限に連続してCPUを占有する事がなくなる。

【0059】第2の発明によれば、同一のリアルタイムプロセスが連続してCPUを占有した回数が所定回数になると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングを実行することにより、同一のリアルタイムプロセスが連続してCPUを占有した場合にユーザが任意の対処を行うことが可能になる。

【0060】第3の発明によれば、同一のリアルタイムプロセスが連続してCPUを占有した回数が所定回数になると、現在進行中のリアルタイムプロセスを強制終了させるとともにそのリアルタイムプロセスをプロセス待ち行列のリンクから外すことにより、異常原因等により連続してCPUを占有したリアルタイムプロセスが除去され、以後正常な動作に復帰する。

【0061】第4の発明によれば、同一のリアルタイムプロセスが連続してCPUを占有した回数が所定回数になると、現在進行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間で互いの優先順位を入れ換えることにより、同一のリアルタイムプロセスが無制限に連続してCPUを占有する事がなくなる。

【0062】第5の発明によれば、リアルタイムクラスのプロセスが連続してCPUを占有した回数が所定回数になると、プロセス待ち行列にリンクされている対話型クラスの中の優先順位が最高の対話型プロセスにCPU占有を割り当てることにより、リアルタイムプロセスが無制限に連続してCPUを占有する事がなくなる。

【0063】第6の発明によれば、リアルタイムクラスのプロセスが連続してCPUを占有した回数が所定回数になると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングを実行することにより、リアルタイムプロセスが連続してCPUを占有した場合にユーザが任意の対処を行うことが可能になる。

【0064】これらのことから各発明について、次のような効果が得られる。

(1) 同一または同一クラスのリアルタイムプロセスの 40

暴走を防止できる。

(2) 一定時間内に処理が終了しない場合でも、計算機システムがデッドロックにおちいることを防止できる。

(3) 同じく、一定時間内に処理が終了しない場合でも、対話型プロセスが全く作動しないという事態が避けられる。

(4) 対話型プロセスから暴走等を起こした同一のリアルタイムプロセスを強制終了させることも可能である。

(5) さらには、同一のリアルタイムプロセスが暴走等を起こしても、それを異常とみなすことなくそのままリアルタイム処理を継続することも可能である。

【図面の簡単な説明】

【図1】本発明が適用されるオペレーティングシステムの構成を示す概念図である。

【図2】第1の発明に係る第1の実施例を示すフローチャートである。

【図3】図2の要部を示すフローチャートである。

【図4】第1の発明に係る第2の実施例を示すフローチャートである。

【図5】第2の発明に係る第3の実施例を示すフローチャートである。

【図6】第3の発明に係る第4の実施例を示すフローチャートである。

【図7】第4の発明に係る第5の実施例を示すフローチャートである。

【図8】第5の発明に係る第6の実施例を示すフローチャートである。

【図9】第5の発明に係る第7の実施例を示すフローチャートである。

【図10】第6の発明に係る第8の実施例を示すフローチャートである。

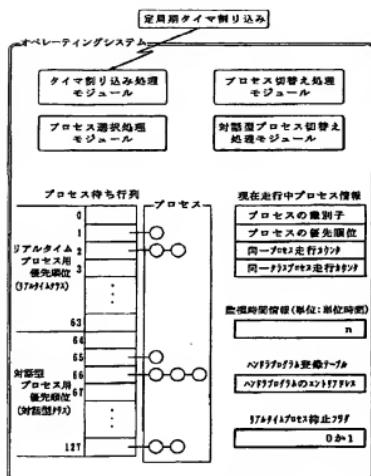
【図11】従来のスケジューリング処理を示すフローチャートである。

【図12】従来のスケジューリング処理を示すフローチャートである。

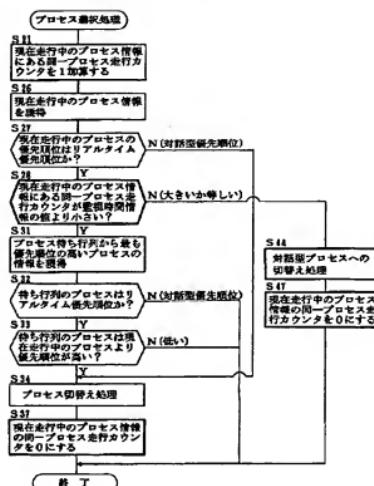
【図13】従来のスケジューリング処理を示すフローチャートである。

【図14】従来のオペレーティングシステムの構成を示した概念図である。

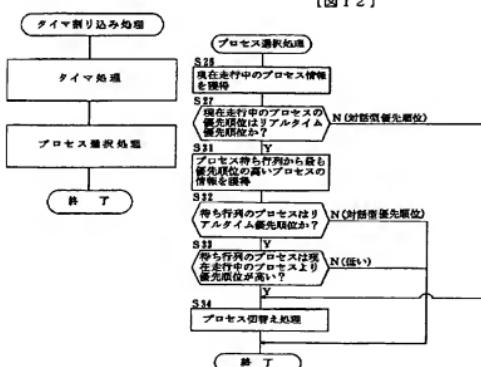
【図1】



【図2】

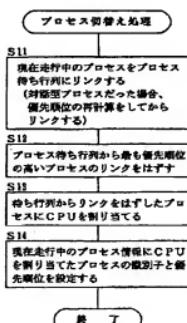


【図11】

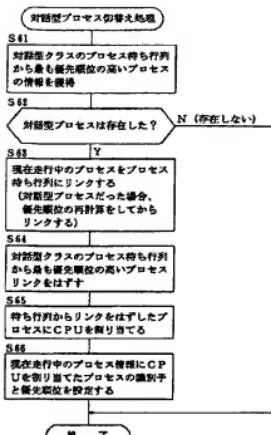


【図12】

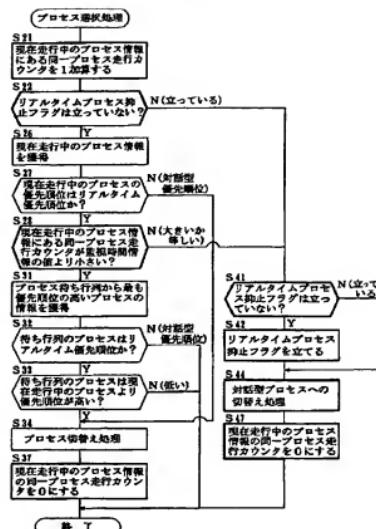
【図13】



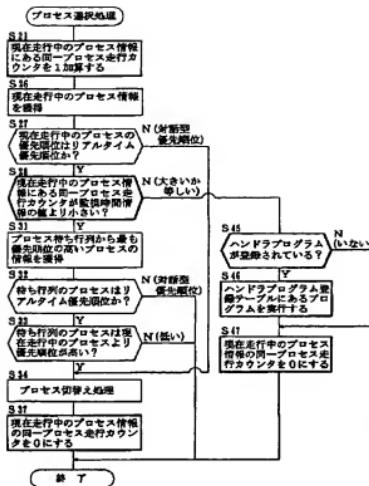
【図3】



【図4】



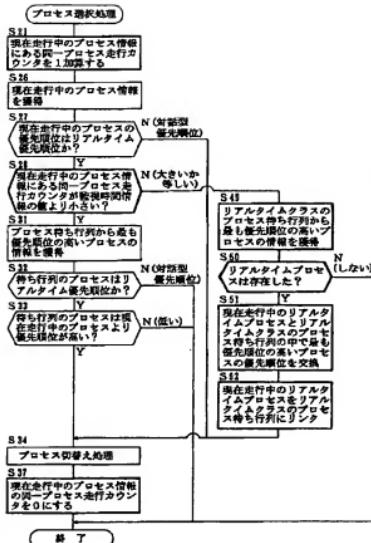
【図5】



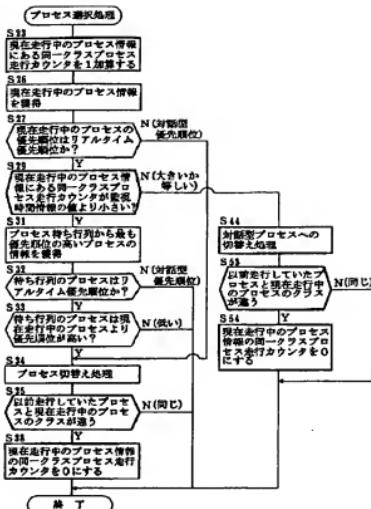
【図6】



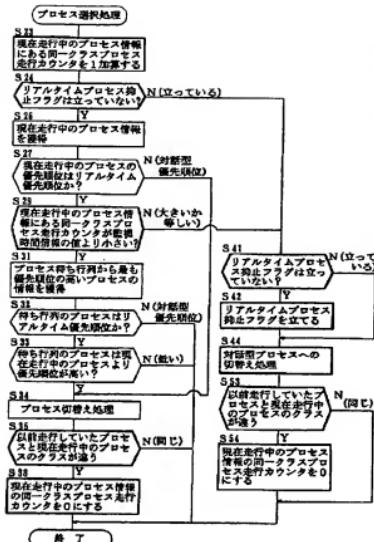
【図7】



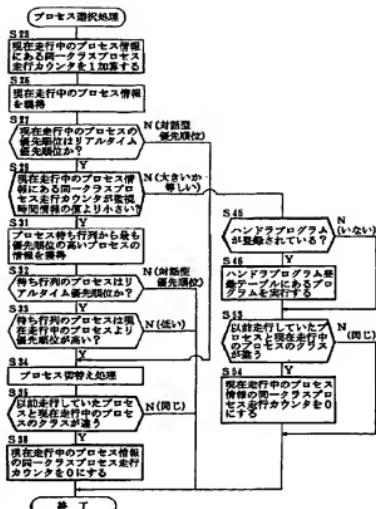
【図8】



(图9)



(図10)



【図14】

